CMSC110 Introduction to Computing

Lab#7
Week of October 17, 2016

Today, we will experiment with arrays. Arrays store a number of data values using a single variable name. All values in an array have to be of the same type.

Part 1: Look at the following commands:

```
int[] n = new int[1000];
for (int i=0; i < n.length; i++) {
    n[i] = 42;
}
```

First, we create an array, called **`n[]`**, to hold 1000 integer values. Next, we are using a loop to store the number 42 in all the 1000 entries in the array. Study the example carefully.

- Try this in Processing. Use `println()` to print out elements of the array to convince yourself that the code above works.
- Next, write the code to initialize an array, called **`m[]`**, to all 0s. Write it in your Processing file, below the code initializing `n`.
- Write a function, called **`allZeros()`**, that checks whether an array of `int`s contains all 0s.
- Now, try the following commands:

  ```
  println(allZeros(n));
  println(allZeros(m));
  ```

  Before running this code, what do you expect to see? Is this indeed what you see? What happens if, after the initialization code you wrote above, you set `m[0] = 13;`?
- The code above creates `n` with `1000` elements. What will happen if you set `n[1000]`? What will happen if you try to print `n[-1]`?

Part 2: The following commands are used to simulate the rolling of a die 10,000 times. The resulting outcomes are stored in an array, called **`outcome`**:

```
int[] outcome = new int[10000];
for (int i=0; i < outcome.length; i++) {

     // fill it up with random values
     outcome[i] = int(random(1,7));
}
```

Study the example above carefully and enter it in Processing. Then, create an array called **`counts`** to store the number of rolls of each of the six faces of a die using the events generated in the previous outcome array. That is, if there were 1,652 rolls of "2", then `counts[2]` would be `1652`. Once `counts` is set up, print out its contents with `println`.

Hint: You may use the following algorithm:

```
counts is an array of integers containing seven entries.
Initialize counts to contain all zeroes (these are initial counts).
For each entry in outcome:
  Increment by 1 the appropriate entry in counts
```

Questions:

Why is the entry for **`counts[0]`** zero?

Based on the contents of the array counts, what can you say about the chances of rolling a 5? Write an expression below that computes this:

Self-Assessment: Array Basics

Arrays are very important in computing and programming. So it is worthwhile reviewing the basics. Answer the questions below as precisely and as concisely as you can. Review your answers with your instructor or a TA. In the following definitions, two arrays are defined:

```
int[] n = new int[10];
String[] names = new String[20];
```

Answer the following questions:

1. What is the name of the integer array defined above? __________
2. How many values can the arrays, **`n[]`** store? __________, **`names[]`**? __________
3. What is the value of expression, **`n.length`**? __________, **`names.length`**? __________
4. Write commands to store the value 1 in all locations of the array, **`n[]`**.
5. Write commands to store the name **"`Donald`"** in all locations of the array, **`names[]`**.
6. What is the index of the first element of the array, **`n[]`**?__________, **`names[]`**? __________
7. What is the index of the last element of the array, **`n[]`**?__________, **`names[]`**? __________
8. Write a command to store the value 23 in the 4th location of the array, **`n[]`**.
9. Write commands to store the name **"`Mike`"** in the first and last locations of the array, **`names[]`**.

10. Write commands to store the names **"Hillary"** and **"Tim"** in alternating locations of the array, **names[]**.

11. The following commands should fill the array, **n[]** with random numbers between 1 and 10 (inclusive). Complete the code below in order to accomplish this.

```
for (int i=0; i < ____________; i++) {

   n[i] = _int(___________________________________);

}
```

12. After the loop below is completed, max should contain the value of the largest element in the array **n[]**:

```
int max = n[0];
for (int i=1; i < n.length; i++) {
   if (_____________________) {
      max = n[i];
   }
}

// print the largest element in n[]
println("The largest number in n[] is: " + max);
```

13. Write commands to determine and print the smallest element in the array, **n[]**:

Part 3 (Home Work): Visualizing Data Using Pie Charts.

Consider the set of numbers in the Table below:

Table. A Set of Sample Values

Petroleum	Coal	Natural Gas	Nuclear	Renewable	Hydropower
40.0	23.0	22.0	8.0	4.0	3.0

Create the two arrays as shown below and observe the results of the two print commands:

```
// Names of various energy sources
String[] energySource = {"Petroleum", "Coal", "Natural Gas",
                         "Nuclear", "Renewable", "Hydropower"};

float[] consumption = {40.0, 23.0, 22.0, 8.0, 4.0, 3.0};

println(consumption.length);
println(consumption);
```

Write commands to print the values from **`energySource`**[] and **`consumption[]`** in the format shown below:

```
Petroleum, 40.0
Coal, 23.0
Natural Gas, 22.0
Nuclear, 8.0
Renewable, 4.0
Hydropower, 3.0
```

Next, enter and run the program below:

```
// Names of various energy sources and consumption
String[] energySource = {"Petroleum", "Coal", "Natural Gas",
                         "Nuclear", "Renewable", "Hydropower"};
float[] consumption = {40.0, 23.0, 22.0, 8.0, 4.0, 3.0};

color[] swatch = {color(247,87,87), color(247,240,87),
                  color(97,87,247), color(247,87,232),
                  color(87,247,88), color(26,11,162)};

void setup() {
  size(500, 500);
  background(255);
  drawPie(consumption, energySource);
} // setup()
```

```
void drawPie(float[] data, String[] legend) {
  int pieX = 200;    // Pie location and size
  int pieY = 200;
  int pieSize = 250;

  float startAngle = 0;  // Pie variables
  float stopAngle = 0;

  int legendX = 350;    // legend location
  int legendY = 150;
  int legendDelta = 26;

  textSize(24);

  for (int i=0; i < data.length; i++) {
    stopAngle = startAngle + data[i]*TWO_PI/100.0;

    fill(swatch[i]);
    noStroke();
    arc(pieX, pieY, pieSize, pieSize, startAngle, stopAngle);

    // Draw legend
    text(legend[i], legendX, legendY);
    legendY += legendDelta;

    // update for next pie
    startAngle = stopAngle;
  }

  fill(0);
  text("Energy Consumption\n% of Energy Sources\n(USA, 2005)", 75,
400);
} // drawPie()
```

Study the program above carefully and make sure you understand it completely. Ask questions!